Appl. Num. Anal. Comp. Math. **1**, No. 1, 104 – 112 (2004) / **DOI** 10.1002/anac.200310009

An advanced variant of an interpolatory graphical display algorithm

G. Ala [*1], **E. Francomano** [**2,3], **A.Tortorici**[2, 3], **E.Toscano** [4], and **F. Viola** [1]

[1] Dipartimento di Ingegneria Elettrica - Universitá degli Studi di Palermo, viale delle Scienze, 90128 Palermo, Italia
[2] Dipartimento di Ingegneria Informatica - Universitá degli Studi di Palermo, viale delle Scienze, 90128 Palermo, Italia
[3] ICAR, Istituto per il CAlcolo e Reti ad alte prestazioni, CNR, viale delle Scienze, 90128 Palermo, Italia
[4] Dipartimento di Fisica e Tecnologie Relative - Universitá degli Studi di Palermo, viale delle Scienze, 90128 Palermo, Italia

Received 30 June 2003, revised 30 October 2003, accepted 2 December 2003
Published online 15 March 2004

Key words Spline functions, graphical display algorithm, computational complexity
Subject classification 65C20, 33F05, 65D07

In this paper an advanced interpolatory graphical display algorithm based on cardinal B-spline functions is provided. It is well-known that B-spline functions are a flexible tool to design various scale rapresentations of a signal. The proposed method allows to display without recursion a function at any desiderable resolution so that only initial data and opportune vectors weight are involved. In this way the structure of the algorithm is independent across the scale and a computational efficiency is reached. In this paper mono and bi-dimensional vectors weight generated by means of centered cubic cardinal B-spline functions have been supplied.

1 Introduction

The general idea of representing a signal at multiple scales is well known. Multiscale representation is of crucial importance in describing the structure of the world at various resolution levels. Efficient scaling mechanisms are needed to allow users to vary the size of images interactively, to concentrate on some details or to get a better overview. Other utilizations include geometrical transformations and image registration where is necessary to resample the image to an undistorted or reference coordinate system. A general framework of scale-space representation is in the context of B-splines [1, 2, 3]. In this paper an interpolatory graphical display algorithm [1] based on the refinable property of the spline functions is taken into account. The algorithm proposed by C. K. Chui is an iteration version of a subdivision scheme and the computational load becomes extremely large, when the scale gets larger. Therefore, an opportune re-arrangement of the scale-space filtering is provided. Namely, a pre-processing stage is performed generating a binary tree of vectors weight regarding the desiderable resolution level. These vectors weight can be directly applied to the initial data set so avoiding the recursion in the computation and noticeably improving the computational complexity. The modified algorithm has been formulated for one and multidimensional spaces [4, 5] and the vectors weight for cubic cardinal centered B-spline functions have been provided for tree levels. When scattered data have to be treated the approximation results can be appreciable improved by using a quasi-interpolant operator. The operator can be applied by opportunely modifying the vectors weight. In the paper the quasi-interpolant operator based on cubic cardinal centered B-spline functions has been used and the modified binary tree has been provided. The paper is organized as follows. Section 2 is devoted to present the interpolatory graphical display algorithm developed by C. K. Chui. Section 3 reports the monodimensional and multidimensional schemes of the recurrence-free variant of the process.

* e-mail: ala@diepa.unipa.it, Phone: +00 39 091 6615288, Fax: +00 39 091 488452
** Corresponding author: e-mail: elisa@cere.pa.cnr.it, Phone: +00 39 091 238266

2 The interpolatory graphical display algorithm

This section is devoted to describe the interpolatory graphical display algorithm proposed by C. K. Chui [1]. Let $B_m(x)$ be a B-spline function of order m [1, 4]. The relation:

$$B_m(x) = \sum_{k=0}^{m} 2^{-m+1} \left(\begin{array}{c} m \\ k \end{array} \right) B_m(2x-k) = \sum_{k=0}^{m} p_{m,k} B_m(2x-k), \tag{1}$$

is known as one of the two-scale relations that give rise to a hierarchical representation of a signal at multiple scales. Given a discrete data set of points $\{a_l^{(j_0)} = f(l \cdot 2^{-j_0})\}_{l,j_0 \in Z}$ and

$$f(x) \approx \sum_{l \in Z} a_l^{(j_0)} B_m(2^{j_0} x - l) \tag{2}$$

the aim is to compute all the values of the sequence:

$$\{f(k \cdot 2^{-j_1})\}_{k,j_1 \in Z} \qquad \forall j_1 \geq j_0, \tag{3}$$

so that to display the graph of $f(x)$ it is adequate to display the sequence (3), when the fixed value j_1 is sufficiently large. First of all, for $j = j_0, ..., j_1 - 1$, the sequence $\{a_l^{(j)}\}_{l \in Z}$ have to be handled by inserting a zero term between two consecutive elements, namely $\{\tilde{a}_l^{(j)}\}_{l \in Z}$ have to be generated where $\tilde{a}_{2k}^{(j)} = a_k^{(j)}$ and $\tilde{a}_{2k+1}^{(j)} = 0$. Hence, by considering:

$$f^{(j)}(x) = \sum_{l \in Z} a_l^{(j)} B_m(2^j x - l), \tag{4}$$

and taking into account the formula (1), the identity $f^{(j+1)}(x) = f^{(j)}(x)$ generates the following relation:

$$a_l^{(j+1)} = \sum_{k \in Z} 2^{-m+1} \left(\begin{array}{c} m \\ l-k \end{array} \right) \tilde{a}_k^{(j)}. \tag{5}$$

Therefore,

$$f(k \cdot 2^{-j_1}) \approx \sum_{l \in Z} a_l^{(j_1)} B_m(k-l). \tag{6}$$

The fundamental steps of the described process are reported in the following.

ALGORITHM 1
Given:
Data set $\{a_l^{(j_0)} = f(l \cdot 2^{-j_0})\}_{l,j_0 \in Z}$ and $j_1 \geq j_0$

Steps:

1. For $j = j_0, ..., j_1 - 1$
   1.1 generation of $\{\tilde{a}_l^{(j)}\}_{l \in Z}$
   1.2 computation of the coefficients $\{a_l^{(j+1)}\}_{l \in Z}$
2. computation of $\{f(k \cdot 2^{-j_1})\}_{k \in Z}$

By supposing to work with n initial data $\{a_l^{(j_0)} = f(l \cdot 2^{-j_0})\}_{l=1}^{n}$, for each level j the computations in 1.2 need $5n2^j$ products. Therefore $5n(2^{j_1} - 1)$ products need to generate all the coefficients up the level j_1. Furthermore, for the computation of each element of the statement 2 need $m - 1$ products.

3 The recurrence-free variant

In this section details of the developed recurrence-free variant of the computational process described in section 2 are reported.

3.1 Monodimensional scheme

As shown in the formula (5) the coefficients are recursively computed by involving the upsampled values of the previous level. This pecularity has been avoided by generating a mapping of the process. Namely, each value of the function at a generic level t is generated by means of a convolution between an opportune vector weight and a subset of the initial data values. The vectors weight are carried out by means of a binary tree, namely:

$\diamond$ the root $v^{(0)}$is a vector of size $m-1$ whose entries are the values of the B-spline function computed in the integer knots of the support (0,m) ;

$\diamond$ the 2^t nodes $v_h^{(t)}$, where $t \geq 1$ and h is a sequence of t binary digits, are generated by means of the tensorial product of the vectors weight of the previous level and $\{p_{m,k}\}_{k=0}^{m}$.

Therefore, at the level $t = 1$, the nodes $v_0^{(1)}$ and $v_1^{(1)}$ are generated starting from the vector $v^{(1)}$ obtained as follows:

$$v^{(1)}(r) = \sum_{k=1}^{m-1} g^{(0)}(k, r-k+1) \qquad r = 1, ..., 2m-1, \tag{7}$$

where:

$$G^{(0)} = v^{(0)} \bigotimes \{p_{m,k}\}_{k=0}^{m}, \tag{8}$$

and $g^{(0)}(i,j) = 0$ for $j \leq 0$. For istance, by considering the centered cardinal B-spline functions of order $m = 4$:

$v^{(0)} = (\frac{1}{6}, \frac{4}{6}, \frac{1}{6}) \quad p_{4,0} = \frac{1}{8} \quad p_{4,1} = \frac{4}{8} \quad p_{4,2} = \frac{6}{8} \quad p_{4,3} = \frac{4}{8} \quad p_{4,4} = \frac{1}{8},$

$$G^{(0)} = v^{(0)} \bigotimes \{p_{m,k}\}_{k=0}^{4} = \frac{1}{6\times 8} \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix},$$

$v^{(1)} = \frac{1}{6\times 8}(1, 8, 23, 32, 23, 8, 1).$

Consequently, the two nodes $v_0^{(1)}$ and $v_1^{(1)}$ of the binary tree are generated by extracting the entries of the vector $v^{(1)}$ located in the position even and odd respectively. At the level $t > 1$, for each node of the previous level the matrix $G_h^{(t-1)}$ is generated:

$$G_h^{(t-1)} = v_h^{(t-1)} \bigotimes \{p_{m,k}\}_{k=0}^{m} \tag{9}$$

and

$$v^{(t)}(r) = \sum_{k=1}^{s} g_h^{(t-1)}(k, r-k+1) \qquad r = 1, ..., s+m, \tag{10}$$

where $s = m$ or $s = m-1$ is the size of the vector $v_h^{(t-1)}$ and $g_h^{(t-1)}(i,j) = 0$ for $j \leq 0$. At this time the vectors weight $v_{0h}^{(t)}$ and $v_{1h}^{(t)}$ can be generated. Namely, $v_{0h}^{(t)}$ is composed by the entries of the vector $v^{(t)}$ with even indeces and $v_{1h}^{(t)}$is composed by the ones with odd indeces. In the Fig.1 the vectors weight regarding the

centered cardinal B-spline functions of order $m = 4$ are reported for $t = 1, 2, 3$. For the sake of simplicity, at each level *t* the common denominator 6×8^t is dropped in order to achieve integer operations. Of course the final result must be divided by the omitted quantity. In the following the outlined algorithm is reported.

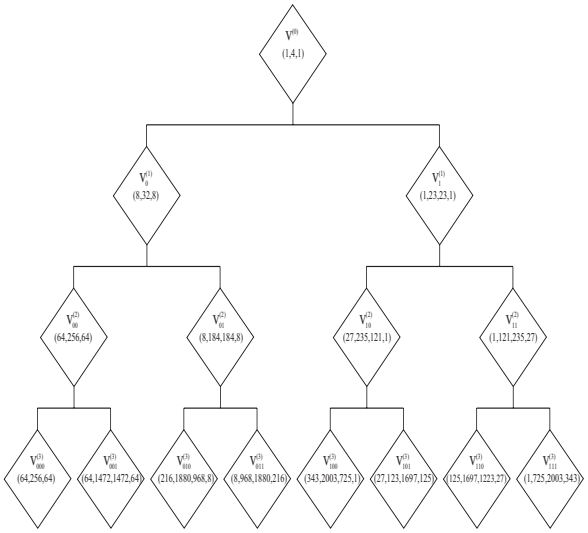

Fig. 1 The binary tree generated by centered cardinal B-spline functions of order $m = 4$.

ALGORITHM 2 (Pre-processing : tree generation)
Given: $m, t_f, v^{(0)}(k) = B_m(k) \quad k = 1, ..., m-1$

Steps:

1. Computation of $\{p_{m,k}\}_{k=0}^{m}$
2. Repeat for $t = 1$,...,t_f:
   - 2.1 if $t = 1$: computation of the formulae (7)-(8)
     else
   - 2.1 computation of the formulae (9)-(10) for each node of the level $t-1$
   - 2.2 $v_{0h}^{(t)} \ (k) = v^{(t)}(2k) \quad k = 1, ..., \lfloor \frac{s+m}{2} \rfloor$,
   - 2.3 $v_{1h}^{(t)}(k) = v^{(t)}(2k-1) \quad k = 1, ..., \lfloor \frac{s+m+1}{2} \rfloor$.

At this time, let $\{f(x_i)\}_{i=1}^{n} = \{f_i\}_{i=1}^{n}$ be the initial data set. The $\{f_i^{(t)}\}_{i=1}^{2^t n}$ values can be generated as follows:

$$f_i^{(t)} = \sum_{k=1}^{m_v} v_h^{(t)}(k) f_{q+k} \qquad i = 1, ..., 2^t n \qquad q = \lfloor \frac{i}{2^t} \rfloor - 2, \tag{11}$$

where $m_v$is the size of the vector $v_h^{(t)}$. The vector $v_h^{(t)}$ involved in the formula (11) is the vector where h is the binary value of r with $r \equiv i \ mod \ 2^t$.
In Figs. 2-3 the initial function values and the vectors weight involved in the computation of some elements $f_i^{(t)}$ are shown.

Therefore, for each element $f_i^{(t)}$ need at most m products and 1 division, so that the computational complexity is independent across the scale. When scattered data are given the approximation results can be appreciable

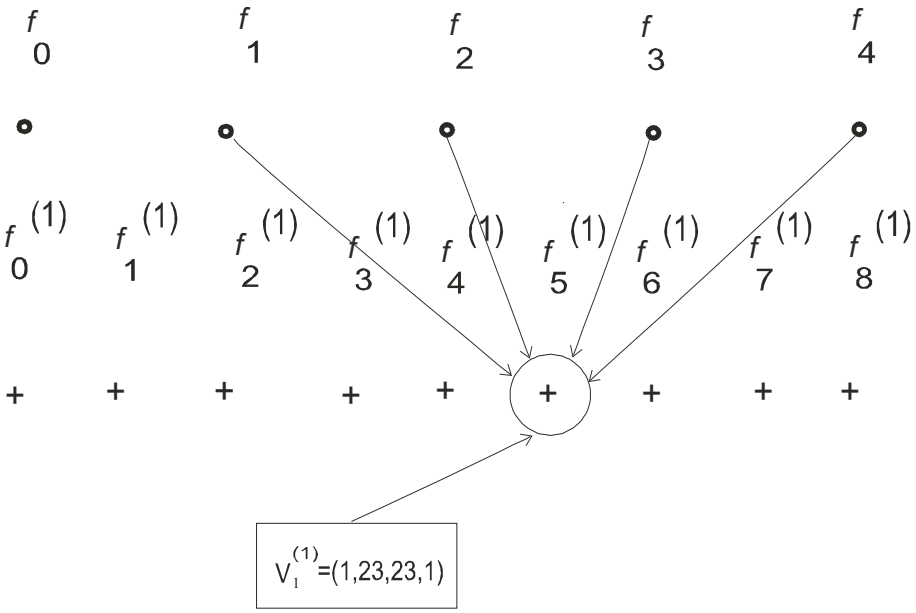

Fig. 2 Generation of $f_5^{(1)}$.

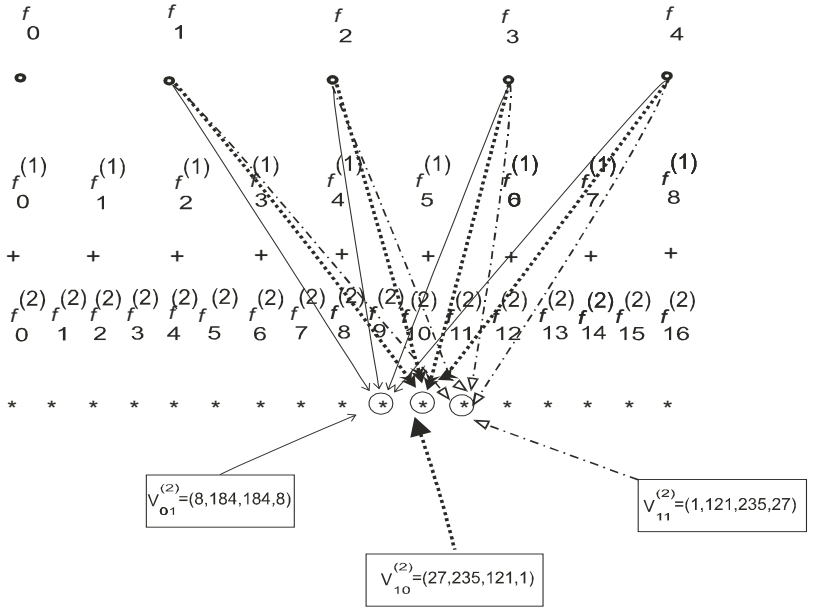

Fig. 3 Generation of $f_9^{(2)}$,$f_{10}^{(2)}$ and $f_{11}^{(2)}$.

improved by applying a quasi-interpolant operator on the initial data set. For istance, by considering the centered cardinal B-spline functions of order $m = 4$ a quasi-interpolant operator is [1]:

$$(Qf)(x) = \sum_{l \in Z} \frac{1}{6}(-f_{l-1} + 8f_l - f_{l+1})B_4(x + 2 - l). \tag{12}$$

Consequently in the standard algorithm the coefficients $\{a_l^{(j_0)}\}_{l,j_0 \in Z}$ are modified as follows:

$$a_l^{(j_0)} = \frac{1}{6}(-f_{2^{-j_0}(l-1)} + 8f_{2^{-j_0}l} - f_{2^{-j_0}(l+1)}). \tag{13}$$

Namely by considering :

$$\omega = (-\frac{1}{6}, \frac{4}{3}, -\frac{1}{6})^T \tag{14}$$

and:

$$S = \{s(i,j)\}_{i=1,...,m_v;j=1,...,3} = v_h^{(t)} \bigotimes \omega, \tag{15}$$

the modified vectors are:

$$\overline{v}_h^{(t)}(r) = \sum_{k=1}^{m_v} s(k, r-k+1) \qquad r = 1, ..., m_v + 2, \tag{16}$$

with $s(i,j) = 0$ for $j \leq 0$. Hence:

$$\overline{f}_i^{(t)} = \sum_{k=1}^{m_v} v_h^{(t)}(k)\frac{1}{6}(-f_{q+k-1}+8f_{q+k}-f_{q+k+1}) = \sum_{k=1}^{m_v+2} \overline{v}_h^{(t)}(k) f_{q+k-1} \quad i = 1, ..., 2^t n \;\; q = \lfloor \frac{i}{2^t} \rfloor - 2. \tag{17}$$

In Fig.4 the binary tree of the modified vectors weight is reported for $t = 1, 2, 3$ by taking into account the centered cardinal B-spline functions of order $m = 4$.

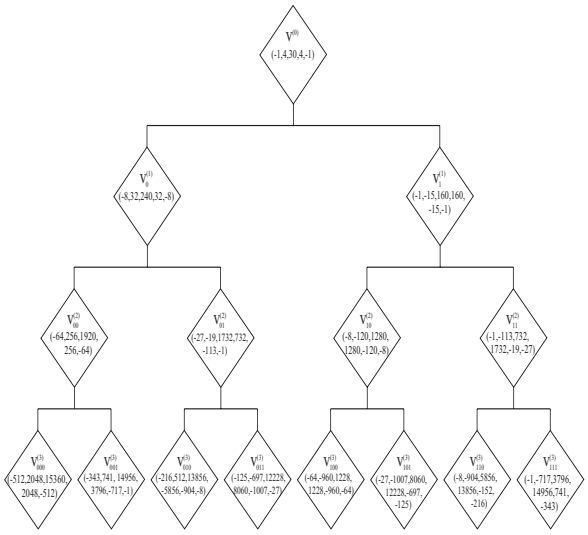

Fig. 4 The binary tree generated by using cardinal centered B-spline functions of order $m = 4$ and the quasi- interpolant operator of formula (12).

3.2 Multidimensional schemes

It is well-known that dealing with functions of more than one variable the standard procedure is the use of the tensor-product of one dimensional spaces of polynomial splines [4, 5]. Therefore, the modified graphical display algorithm in a multidimensional s space can be formulated as follows:

$$f^{(t)}_{i_1,i_2,...,i_d} = \sum_{k_1}\sum_{k_2}...\sum_{k_d} v^{(t)}_{h_1} \bigotimes v^{(t)}_{h_2} \bigotimes ... \bigotimes v^{(t)}_{h_s}(k_1,k_2,...,k_d) f_{q_1+k_1,q_2+k_2,...,q_d+k_d} \qquad (18)$$

$$q_1 = \lfloor \tfrac{i_1}{2^t} \rfloor - 2, ..., q_d = \lfloor \tfrac{i_d}{2^t} \rfloor - 2.$$

In the following will refer to 2D space and in order to better explain the algorithm, the computations of some elements are reported. By referring to the vectors weight relative to $t = 1$ of Fig.1 the following four 2D vectors weigth are generated:

$$v^{(1)}_0 \bigotimes v^{(1)}_0 = \frac{1}{6^2 \times 8^2} \begin{pmatrix} 64 & 256 & 64 \\ 256 & 1024 & 256 \\ 64 & 256 & 64 \end{pmatrix}, \qquad (19)$$

$$v^{(1)}_0 \bigotimes v^{(1)}_1 = (v^{(1)}_1 \bigotimes v^{(1)}_0)^T = \frac{1}{6^2 \times 8^2} \begin{pmatrix} 8 & 184 & 184 & 8 \\ 32 & 736 & 736 & 32 \\ 8 & 184 & 184 & 8 \end{pmatrix}, \qquad (20)$$

$$v^{(1)}_1 \bigotimes v^{(1)}_1 = \frac{1}{6^2 \times 8^2} \begin{pmatrix} 1 & 23 & 23 & 1 \\ 23 & 529 & 529 & 23 \\ 23 & 529 & 529 & 23 \\ 1 & 23 & 23 & 1 \end{pmatrix}, \qquad (21)$$

and the formula (18), for instance, for the value $f^{(1)}_{5,7}$ can be expressed as:

$$f^{(1)}_{5,7} = \sum_{k_1=1}^{4}\sum_{k_2=1}^{4} v^{(1)}_1 \bigotimes v^{(1)}_1(k_1,k_2) f_{k_1,1+k_2} = \frac{1}{6^2 \times 8^2}[(f_{1,2} + f_{1,5} + f_{4,2} + f_{4,5}) + 23(f_{1,3} + f_{1,4} + f_{2,2} + f_{2,5} + f_{3,2} + f_{3,5} + f_{4,3} + f_{4,4}) + 529(f_{2,3} + f_{2,4} + f_{3,3} + f_{3,4})]. \qquad (22)$$

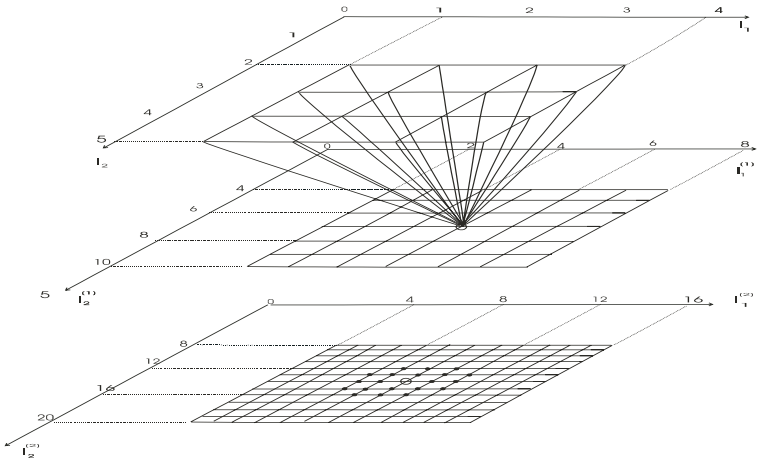

Fig. 5 Generation of $f^{(1)}_{5,7}$.

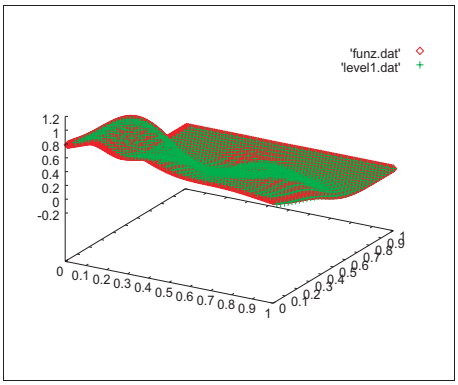

Fig. 6 (Colour online at www.interscience.wiley.com) Function (24) and approximated function obtained with one level of the described process composed with the quasi- interpolant operator.

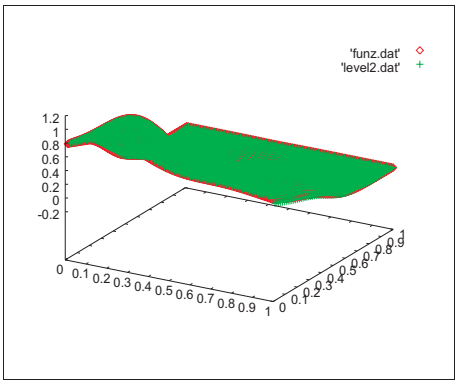

Fig. 7 (Colour online at www.interscience.wiley.com) Function (24) and approximation result obtained with two levels of the described process composed with the quasi-interpolant operator.

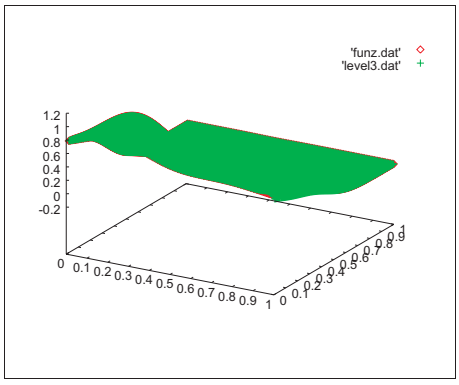

Fig. 8 (Colour online at www.interscience.wiley.com) Function (24) and approximation result obtained with three levels of the described process composed with the quasi- interpolant operator.

The same set of initial data, or a subset, involved in $f^{(1)}_{5,7}$ is also interested in the generation of the values $f^{(1)}_{5,6}, f^{(1)}_{4,7}, f^{(1)}_{6,7}, f^{(1)}_{5,8}$. The same values, or a subset, are interested at the level 2 for generating the elements pointed out in Fig.5.

As just underlined in 1D it is possible modify the final vectors weight in order to obtain smoothness data and improving the final results by means of a quasi-interpolant operator. Therefore, the modified vectors weight are carried out by means of the tensorial products and the quasi-interpolant operator derived from the formula (12). In the following is reported the computation for the previously analyzed value $f_{5,7}^{(1)}$:

$$\begin{aligned}\overline{f}_{5,7}^{(1)} = \frac{1}{6^4 \times 8^2}[&(f_{0,1} + f_{5,1} + f_{0,6} + f_{5,6}) + 15(f_{1,1} + f_{4,1} + f_{0,2} + f_{3,2} + f_{0,5} \\ &+f_{5,5} + f_{1,6} + f_{4,6}) - 160(f_{2,1} + f_{3,1} + f_{0,3} + f_{5,3} + f_{0,4} + f_{5,4} + f_{2,6} \\ &+f_{3,6}) + 225(f_{1,2} + f_{4,2} + f_{1,5} + f_{4,5}) - 2400(f_{2,2} + f_{3,2} + f_{1,3} + f_{4,3} \\ &+f_{1,4} + f_{4,4} + f_{2,5} + f_{3,5}) + 25600(f_{2,3} + f_{3,3} + f_{2,4} + f_{3,4})]. \end{aligned} \tag{23}$$

Of course more elements than formula (22) are involved in the new computation (23). Namely, for $\overline{f}_{i,j}^{(t)}$ need $(m_v + 2)^2$ values of the function f.
In Figs. 6, 7, 8 the results achieved by applying to the test function [8] (24) three levels of the algorithm composed with the quasi-interpolant operator described and by using a discretization step $h = 0.0625$, are presented:

$$\begin{aligned} f(x,y) = 0.75exp[-\frac{(9x-2)^2 + (9y-2)^2}{4}] + 0.75exp[-(\frac{(9x+1)^2}{49} + \frac{(9y+1)^2}{10})] \\ +0.5exp[-\frac{(9x-7)^2 + (9y-3)^2}{4}] - 0.2exp[-(9x-4)^2 + (9y-7)^2], \end{aligned} \tag{24}$$

$x, y \in [0,1] \times [0,1]$.

4 Conclusion

In this paper a fast algorithm for interpolating data by means of B-spline functions is provided. The main result of the computational scheme can be expressed in terms of vectors weight so avoiding the recurrence structure of the subdivision process. The major advantage of the algorithm is in the reduction of the computational complexity which is independent across the scale. The new scheme is proposed for univariate and multivariate splines.

References

[1] C. K. Chui, An introduction to wavelets, Academic Press, (1992).
[2] M. Unser, A. Aldroubi, M.Eden, Fast B-spline transforms for continuos image representation and interpolation, IEEE Transactions on Pattern Analysis and Machine Intelligence, **13**, 277(1991).
[3] Y.-P. Wang, S.L.Lee, Scale-space derived from B-splines, IEEE Transactions on Pattern Analysis and Machine Intelligence, **20**, 1040(1998).
[4] L. Schumaker, Spline Functions: Basic Theory, Wiley–Interscience, 1980.
[5] C. K. Chui, Multivariate splines, SIAM, 1988.
[6] C. K. Chui, Wavelets: a mathematical tool for signal analysis, SIAM, 1997.
[7] C. De Boor, A pratical guide to splines, Springer Verlag, 1978.
[8] R. Franke, G.M. Nielson, Smooth interpolation and large scattered data, International Journal for Numerical Methods in Engineering **38**, 1691(1980).